TAGS: TOPOLOGY-AWARE GRAPH SCHEDULING OF HETEROGENEOUS ROBOTS FOR DELIVERY AND COLLABORATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Heterogeneous robots serve capacity-limited delivery requests and synchronized collaborative tasks by coordinating repeated cargo trips and capability-covering teams over a shared travel network. These operations compete for robot availability: a delivery can delay team formation, while a partial team reserves robots needed for cargo transport. We introduce Topology-Aware Graph Scheduling (TAGS), a unified learning framework that couples delivery progress and partial-team commitments through an event-driven representation of shared robot availability. On this evolving state, a topology-aware entity graph relates available robot resources to task requirements and travel costs, enabling an autoregressive robot–command policy to coordinate cargo dispatch and team formation. We compare TAGS with adapted attention-based learning, mathematical optimization, neighborhood search, and rule-based scheduling on benchmarks with 15, 35, and 50 heterogeneous robots. With 50 robots scheduling 25 delivery and 25 collaborative tasks, best-of-16 TAGS completes every instance and reduces mean makespan by 6.9% and collaborative waiting by 27.1% relative to the respective best full-completion baselines. It retains the lowest mean makespan under zero-shot transfer from 50% to 75% and 100% collaboration; a three-robot demonstration illustrates physical execution.

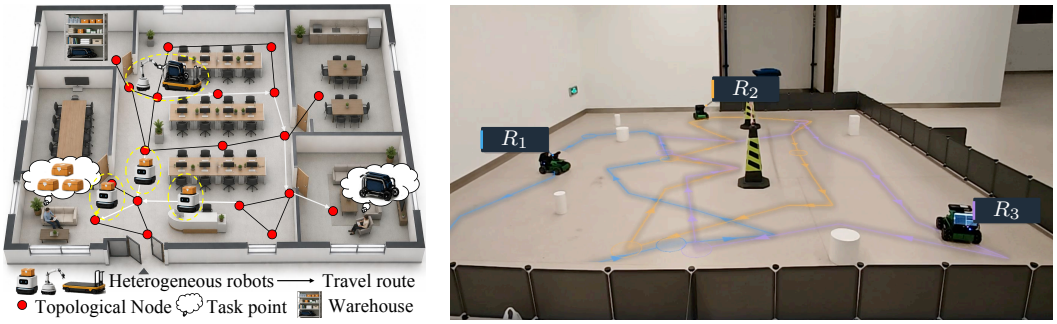
1 INTRODUCTION

Heterogeneous robot teams provide retrieval-and-delivery services in environments connected by corridors or roads. Their workloads can combine divisible delivery requests with tasks requiring several robots to work together, as illustrated in Figure 1. Payload limits require repeated delivery trips, while complementary capabilities determine feasible collaborative teams (François et al., 2016; Park et al., 2026). Both operations occupy the same robots and depend on where compatible supplies and prospective teammates are located. Dispatching a robot to deliver cargo therefore changes the availability of future teams, while assembling a team can temporarily remove robots from delivery service. **How can repeated deliveries and synchronized collaboration be jointly scheduled across a shared team of heterogeneous robots?**

Existing methods offer complementary scheduling approaches. Optimization and neighborhood-search methods explicitly represent routing, capacity, repeated visits, and synchronization (Drexl, 2012; Grangier et al., 2016; Hà et al., 2020). Attention-based routing constructs solutions autoregressively (Kool et al., 2019; Berto et al., 2025); MAPT generates relation-aware joint decisions for dynamic pickup-and-delivery (Zou et al., 2026). Learning-based task-allocation methods use robot and task representations to select assignments and execution sequences (Paul et al., 2022; Wang et al., 2022), while heterogeneous coalition scheduling explicitly tracks capability requirements and synchronized execution (Dai et al., 2025). Joint retrieval-and-delivery scheduling requires tracking delivered and dispatched demand, compatible storage locations, and robots committed to incomplete teams. These dependencies motivate joint scheduling of cargo delivery and team formation over shared robot resources.

We introduce Topology-Aware Graph Scheduling (TAGS) for this joint scheduling problem. An event-driven state tracks delivery progress and collaborative commitments, allowing each robot–

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Illustrative scenario

(b) Physical demonstration

Figure 1: Illustrative scenario and physical demonstration setup. (a) Heterogeneous robots perform repeated delivery and synchronized collaboration over a travel network. (b) Three wheeled robots serve as physical proxies for agents with different capacities and capability profiles configured in software. TAGS allocates tasks online; colored overlays indicate planned routes.

command decision to update the availability of subsequent deliveries and teams. An entity graph then combines the current robot, task, and storage states with shortest-path travel times, so the policy can condition assignments on both resource status and spatial cost. The decoder selects a robot and its command autoregressively, using state-dependent masks to enforce command conditions and an auxiliary waiting penalty to guide synchronization during training. Our contributions are:

- We formulate joint delivery and collaborative scheduling as an event-driven decision process that connects in-transit demand, storage selection, and capability-covering commitments through shared robot availability.
- We develop TAGS, an entity-graph policy that uses topology-dependent travel information to learn robot selection and conditional delivery, collaboration, or storage commands.
- Experiments establish the method's scheduling-quality and runtime trade-offs across three benchmark sizes, isolate its robot-selection and representation components, and demonstrate zero-shot transfer to higher collaborative-task ratios.

2 Related Work

Multi-robot task allocation and scheduling. MRTA coordinates task assignments and execution sequences under robot capabilities, task dependencies, and temporal constraints (Gerkey & Matarić, 2004; Korsah et al., 2013). Market-based methods organize assignments through bids and consensus (Dias et al., 2006; Choi et al., 2009; Quinton et al., 2023). Learning-based approaches encode robot–task relations with capsule attention, warehouse-specific attention, heterogeneous graph attention, or learned matching scores (Paul et al., 2022; Agrawal et al., 2023; Wang et al., 2022; Paul et al., 2024). Event-driven and asynchronous reinforcement learning organizes decisions around macro-action completion and agent availability (Menda et al., 2019; Xiao et al., 2022). E-ITAGS combines learned capability–efficacy maps with interleaved allocation, scheduling, and path planning under time budgets (Liu et al., 2026).

Capacity-constrained retrieval and delivery. Vehicle routing and pickup-and-delivery models express payload limits, request locations, and transport sequences (Laporte, 1992; Berbeglia et al., 2007). Multi-trip, multi-depot, and two-echelon methods additionally model repeated visits and replenishment, using mathematical optimization and neighborhood search (François et al., 2016; Grangier et al., 2016; Li et al., 2020; Vieira et al., 2023). Attention-based policies learn route construction (Kool et al., 2019), and PARCO coordinates parallel agent decisions (Berto et al., 2025). MAPT combines relation-aware entity representations with autoregressive joint-action generation for dynamic pickup-and-delivery (Zou et al., 2026). Capacitated multi-agent pickup-and-delivery also integrates assignment with collision-free path planning (Chen et al., 2021). This literature connects cargo capacity and repeated service to vehicle assignment, replenishment decisions, and route construction across spatially distributed pickup and delivery locations.

Heterogeneous robot collaboration. Capability-aware collaboration forms teams whose collective skills meet task requirements (Rizk et al., 2019; Zhang & Parker, 2013). GRSTAPS integrates trait-based allocation with task planning, motion planning, and scheduling (Messing et al., 2022). TRAITS jointly optimizes team allocation, capability provisioning, and scheduling under resource, battery, and temporal constraints (Park et al., 2026). Reinforcement learning has been used for collective transport, dynamic coalition formation among homogeneous robots, and heterogeneous capability-covering teams (Paul et al., 2023; Dai et al., 2024; 2025). Other approaches study heterogeneous agent policies and qualification-constrained synchronized routing (Bettini et al., 2023; Bendazzoli et al., 2024). These studies characterize team composition and synchronization; coupling those decisions to cargo progress and repeated retrieval introduces an additional resource-allocation interaction for joint scheduling.

3 METHOD

3.1 PROBLEM FORMULATION

We consider a joint retrieval-and-delivery scheduling problem in which heterogeneous robots perform capacity-limited deliveries and synchronized collaborative tasks. The environment is a weighted graph $\mathcal{G}$ whose shortest-path distances determine travel times. The robot set is $\mathcal{R} = \{1, \dots, N\}$, the task set is $\mathcal{T} = \mathcal{T}^D \cup \mathcal{T}^C$ with M tasks, and $\mathcal{W}$ contains storage nodes. Delivery and collaborative tasks form disjoint sets, and all tasks are available initially. Task j is located at p_j and has compatible storage nodes $\mathcal{W}_j \subseteq \mathcal{W}$. Robot i has payload capacity κ_i, compatibility $\mathbf{g}_i \in \{0,1\}^K$ for K delivery-task types, and capabilities $\mathbf{c}_i \in \{0,1\}^C$ over C capability dimensions; an entry of one indicates a supported type or available capability. Storage inventory is unlimited.

Each delivery task $j \in \mathcal{T}^D$ has type $k_j \in \{1, \dots, K\}$ and integer demand m_j. A compatible robot, satisfying $g_{i,k_j} = 1$, retrieves cargo from a node in $\mathcal{W}_j$ and transports an integer quantity up to its capacity to p_j. Demand can be split across trips and robots. Each collaborative task $j \in \mathcal{T}^C$ requires a team S_j satisfying $\sum_{i \in S_j} \mathbf{c}_i \succeq \mathbf{b}_j$ componentwise, where the nonzero vector $\mathbf{b}_j \in \{0,1\}^C$ marks each required capability with a one. Team members assemble at one compatible storage node, depart together, and remain committed until their common completion time. Collaborative tasks use one synchronized transport; delivery quantities apply to the delivery-task class. Each robot executes at most one trip or commitment at a time. Both task classes compete for robot availability: a delivery delays the robot's next commitment, while an incomplete team reserves its members until joint completion. For task completion times C_j, the objective is

$$\min C_{\max}, \qquad C_{\max} = \max_{j \in \mathcal{T}} C_j. \tag{1}$$

3.2 EVENT-DRIVEN ROBOT–COMMAND SCHEDULING

We model scheduling as an event-driven Markov decision process with state space $\mathcal{S}$, action space $\mathcal{A}$, and discount γ (Figure 2). At epoch n and physical time t_n, TAGS selects a_n from s_n; the simulator updates the state and reward r_n, advancing time when no command remains.

State. To track how delivery progress and team commitments affect robot availability, we organize each state into static attributes, robot status, and task progress. The static component records the environment and entity attributes:

$$s^{\text{env}} = \big(\mathcal{G}, \mathcal{W}, \{(\kappa_i, \mathbf{g}_i, \mathbf{c}_i)\}_{i \in \mathcal{R}}, \{(p_j, \mathcal{W}_j, k_j, m_j)\}_{j \in \mathcal{T}^D}, \{(p_j, \mathcal{W}_j, \mathbf{b}_j)\}_{j \in \mathcal{T}^C}\big). \tag{2}$$

Robot states track location, destination, event timing, status, and load. Scalar and vector attributes below refer to the current state s_n, with their time indices omitted:

$$s_n^{\text{robot}} = \{(p_i, p_i^{\text{dest}}, \delta_i, \sigma_i, L_i)\}_{i \in \mathcal{R}}, \tag{3}$$

where p_i is the last reached topology node, p_i^{dest} the destination, and L_i the current delivery load. The status σ_i distinguishes idle, traveling, waiting, and executing robots; δ_i records the time remaining to a scheduled arrival or release. Idle robots and incomplete teams have availability determined by status and commitments until a timed event is scheduled. During transit, graph distances use p_i

until arrival; the destination and countdown retain travel progress. We also record delivery progress and collaborative commitments in the task state:

$$s_n^{\text{task}} = \big(\{(m_j^{\text{rem}}, m_j^{\text{tr}})\}_{j\in\mathcal{T}^D}, \{(S_{j,n}, \omega_j, \sigma_j)\}_{j\in\mathcal{T}^C}\big), \tag{4}$$

where m_j^{rem} is all undelivered demand, including the in-transit amount m_j^{tr}. The current commitment set $S_{j,n}$ grows into the execution team S_j once its capabilities cover the requirement. The storage lock ω_j is unset before the first commitment and then identifies the team's assembly storage. Task status σ_j distinguishes unstarted tasks, team assembly, synchronized transport, and completion. The remaining capability requirement is $\mathbf{b}_j^{\text{rem}} = [\mathbf{b}_j - \sum_{i\in S_{j,n}} \mathbf{c}_i]_+$, where $[\cdot]_+$ clips each component at zero. Combining these components gives

$$s_n = (t_n, s^{\text{env}}, s_n^{\text{robot}}, s_n^{\text{task}}) \in \mathcal{S}. \tag{5}$$

Action. We represent each decision as $a_n = (i_n, y_n)$, pairing the selected robot i_n with its command y_n. A command y dispatches a delivery $\mathrm{D}(j, q)$, commits to collaboration $\mathrm{C}(j)$, or moves to storage $\mathrm{W}(w)$. Delivery quantities are integers $q \in \{1, \ldots, q_{\max}\}$, where $q_{\max}$ bounds the decoder's quantity vocabulary, giving the action space

$$\mathcal{A} = \mathcal{R} \times \big(\{\mathrm{D}(j, q) : j \in \mathcal{T}^D,\ 1 \le q \le q_{\max}\} \cup \{\mathrm{C}(j) : j \in \mathcal{T}^C\} \cup \{\mathrm{W}(w) : w \in \mathcal{W}\}\big). \tag{6}$$

Commands apply to idle, unladen, uncommitted robots. Delivery from the current storage node p_i requires $p_i \in \mathcal{W}_j$, $g_{i,k_j} = 1$, and $q \le \min\{\kappa_i, m_j^{\text{rem}} - m_j^{\text{tr}}\}$. Subtracting in-transit cargo prevents concurrent dispatches from assigning the same demand. Collaborative commitment requires a compatible current storage, agreement with any existing storage lock, and $\mathbf{c}_i^\top \mathbf{b}_j^{\text{rem}} > 0$, indicating at least one capability still needed by an unfinished task. The constructive policy uses a work-conserving dispatch rule: a robot selects among service commands available at its current storage; otherwise it selects a move to another storage that enables service. The current action set is

$$\mathcal{A}_n = \{(i, y) \in \mathcal{A} : \text{command conditions and dispatch rule hold in } s_n\}. \tag{7}$$

Masks exclude robots without commands, then inadmissible robot–command pairs.

Transitions. After each command, the simulator updates delivery progress and team commitments. A delivery dispatch increases m_j^{tr} by q; arrival reduces both m_j^{rem} and m_j^{tr} by q. The first collaborative commitment locks the team's assembly storage, and later commitments update its remaining capability requirement. Members wait from commitment until the team covers that requirement, then depart synchronously and share a completion and release event. Decisions repeat at the same physical time while $\mathcal{A}_n$ is nonempty. The simulator then advances to the next finite event; episodes end on task completion, timeout, or lack of a valid continuation.

Reward. To encourage early completion and reduce team-assembly waiting, we combine a terminal makespan penalty with an incremental waiting penalty:

$$r_n = r_n^{\text{term}} - \beta(W_{n+1} - W_n), \tag{8}$$

where r_n^{term} is $-C_{\max}$ for completion, $-T_{\max}$ for unsuccessful termination, and zero otherwise; $T_{\max}$ is the execution time limit. The weight $\beta \ge 0$ controls the waiting penalty. Starting from $W_0 = 0$, W_n accumulates all collaborative waiting through epoch n, including elapsed portions of ongoing waits and retaining past waiting after teams depart. Its increment charges the time for which partial teams reserve robots while awaiting complementary capabilities. With $\gamma = 1$, successful episodes receive return $-C_{\max} - \beta W_{\text{final}}$, where W_{final} is total waiting at completion.

Policy and training. We condition command selection on the chosen robot by factorizing the joint policy into robot-selection and conditional-command distributions:

$$\pi_\theta(i, y \mid s_n) = \pi_\theta(i \mid s_n)\, \pi_\theta(y \mid s_n, i), \tag{9}$$

where θ denotes policy parameters; the first factor selects an admissible robot and the second selects its command, each normalized over the choices admitted by $\mathcal{A}_n$. The selected pair executes as one environment action. Learned robot selection determines which available capacity or capability to use next; the conditioned command assigns delivery, collaboration, or storage movement.

PPO (Schulman et al., 2017) optimizes the sum of the robot and conditional-command log-probabilities using a separately parameterized value network.

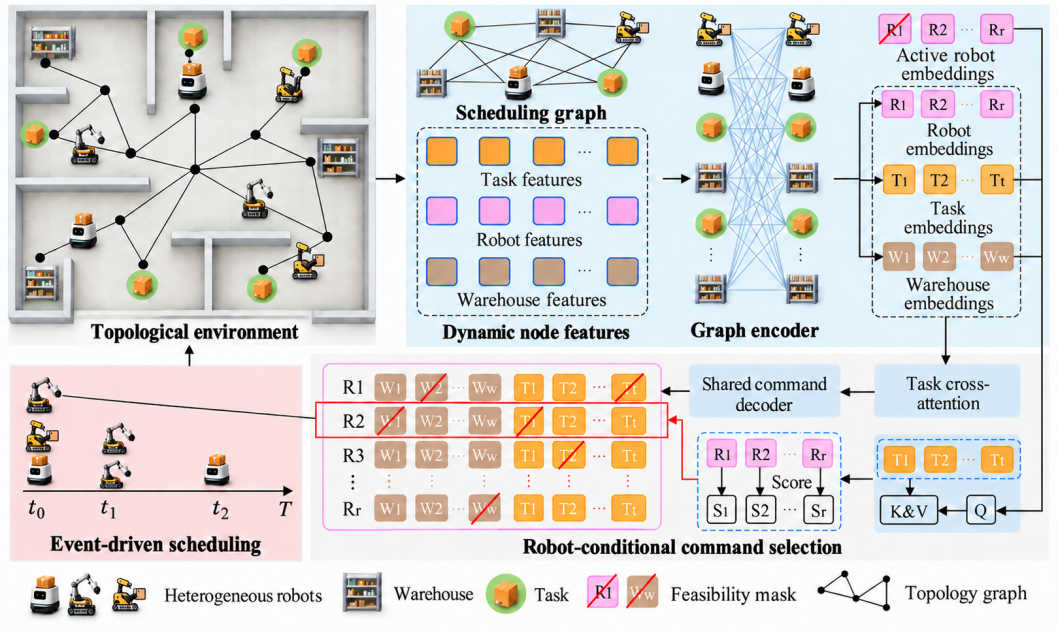


Figure 2: TAGS maps topology-aware robot, task, and storage embeddings to masked robot–command decisions. Delivery dispatches capacity-feasible loads over repeated trips; collaboration assembles capability-covering teams for synchronized execution; storage moves enable further service. Execution updates cargo progress, commitments, and robot availability for the next decision.

3.3 Topology-Aware Graph Policy

As shown in Figure 2, TAGS maps the state components to a complete graph over robot, task, and storage entities, indexed by u, v. All network quantities refer to the current state s_n, with time indices omitted. Each pair receives its shortest-path travel time d_{uv} from the environment topology, allowing attention to relate entities at nonadjacent travel-network nodes. Type-marked features encode capacities, capabilities, delivery and collaboration progress, and execution status. We append each entity's mean distance from robots with available commands, conditioning the shared encoding on the available team before robot selection. The projected features pass through graph-attention layers (Veličković et al., 2018) with Graphormer-inspired travel-time biases (Ying et al., 2021). For one layer and attention head, omitting their indices, the attention weight is

$$\alpha_{uv} = \mathrm{softmax}_v\left(\frac{\mathbf{q}_u^\top \mathbf{k}_v}{\sqrt{d_h}} + b(d_{uv})\right), \tag{10}$$

where $\mathbf{q}_u$ and $\mathbf{k}_v$ are query and key projections of entity embeddings, d_h is their dimension within a head, and softmax normalizes over all entities v in the instance. The affine bias b is learned separately for each layer and head. Each entity thus aggregates robot availability, task requirements, and storage information with travel-dependent weights.

To realize Equation (9), we pool admissible-robot embeddings with padding-masked task and entity means, refine the shared context by task cross-attention, and score available robots. For command selection, the same context construction uses the selected robot's embedding in place of the admissible-robot mean, yielding $\mathbf{z}_i$ (Appendix B.3). For encoded task embedding $\mathbf{h}_j$, the unnormalized score for delivering quantity q from robot i to task j is

$$e_{i,j,q} = \mathrm{ptr}(\mathbf{z}_i, \mathbf{h}_j) + \lambda_d d_{ij} + \left[W_{\mathrm{qty}}(\mathbf{h}_j + \mathbf{z}_i)\right]_q, \tag{11}$$

where ptr is a scaled dot-product pointer score (Vinyals et al., 2015; Kool et al., 2019), d_{ij} is the robot's travel time to task j, and λ_d is a learned scalar. The learned matrix W_{qty} maps the sum of the equal-dimensional task and context embeddings to $q_{\max}$ quantity scores; $[\cdot]_q$ selects the qth entry. The encoder bias shapes interactions among all entities, while the readout distance term ranks

Table 1: Training instance-generation settings. $U_{\mathbb{Z}}[a, b]$ and $U[a, b]$ denote discrete and continuous uniform distributions, respectively.

Parameter	Training range	Parameter	Training range
Tasks M	$U_{\mathbb{Z}}[15, 50]$	Delivery demand m_j	$U_{\mathbb{Z}}[2, 6]$
Storage nodes $\lvert\mathcal{W}\rvert$	$U_{\mathbb{Z}}[2, 5]$	Robot capacity κ_i	$U_{\mathbb{Z}}[1, 3]$
Robots N	Five species, 3–10 each	Edge weight	$U[1, 5]$

targets for the selected robot; the quantity term differentiates feasible loads for each delivery target. Collaborative commands encode a commitment in the first quantity slot; storage commands omit the quantity term and use the corresponding storage embedding and robot-specific distance, with the same λ_d. Masked softmax over robot scores gives $\pi_\theta(i \mid s_n)$; over the selected robot's command scores it gives $\pi_\theta(y \mid s_n, i)$, with both masks defined by Equation (7).

Training and schedule construction use centralized access to robot, task, and storage states. The policy parameters are shared across entities, and an independent critic uses the same graph-context architecture with a scalar value head. At evaluation, frozen policies construct B candidate schedules through the same event-driven transitions; best-of-B decoding selects the completed candidate with minimum makespan. Appendix B specifies layer dimensions and readout implementation.

4 Experiments

The experiments assess joint scheduling, policy design, candidate sampling, and reuse across task mixtures: **Q1:** How does TAGS balance scheduling quality and runtime across problem sizes? **Q2:** How do robot-selection design, topology encoding, and waiting regularization affect scheduling performance? **Q3:** What quality–runtime trade-off does candidate sampling provide? **Q4:** Can a policy trained on mixed delivery and collaborative tasks retain scheduling quality as the collaboration ratio increases, without retraining?

4.1 Experimental Setup

Benchmark. We construct a synthetic joint-scheduling benchmark inspired by HeteroMRTA (Dai et al., 2025). Robots have heterogeneous capacities and five-dimensional, two-hot collaborative capabilities; every task requirement is coverable by the team. All robots support three uniformly sampled delivery types. Tasks have nonempty compatible-storage subsets and occupy distinct nodes from storage. Connected random topologies combine a spanning tree with degree augmentation to 2–4; robots start at storage and travel along shortest paths at unit speed. Loading and service times are zero. Appendix D specifies the generator.

Training and evaluation. Training uses the ranges in Table 1, 50% collaborative tasks, and quantity vocabulary $q_{\max} = 3$. All learned methods use PPO with $\gamma = 1$, Adam learning rate 10^{-4}, and waiting weight $\beta = 0.1$ (Appendix B.5). Three seeds train online for 160,500 steps; final checkpoints are evaluated on 40 independent fixed instances with 15, 35, or 50 robots and equally many tasks (D15/D35/D50), with horizon $T_{\max} = 500$. TAGS requires about 3.5 GPU-hours per seed on one RTX 4090. Main comparisons use best-of-16 decoding; transfer keeps parameters frozen. The anonymous artifact provides the generator, instances, and implementation.

Baselines. *HeteroMRTA* (Dai et al., 2025) adapts separate robot/site encoders and cross-attention to our shared states, commands, masks, PPO setup, and parameter budget, with uniform robot selection. *MIP* and *CP-SAT* use slot-indexed synchronized-routing formulations (Drexl, 2012; Malik et al., 2025; Bendazzoli et al., 2024; Hà et al., 2020); CP-SAT uses OR-Tools (Google OR-Tools, 2024). They start cold with eight activities per robot and 300/900/1200-second budgets on D15/D35/D50; caps of 10 or 12 gave no finite-budget improvement. *ALNS* adapts synchronized insertion and cooperative team-orienteering search (Bendazzoli et al., 2024), using three seeds and the same time budgets. *Greedy* is a handcrafted heuristic within our event-driven framework, sharing the robot–command interface and action masks with the learned policies. Replay checks shared execution constraints; Appendix C details the implementations.

Table 2: Scheduling with 15, 35, and 50 robots and tasks (D15/D35/D50): mean ± population standard deviation across 40 instances per size after seed averaging. Bold/underlined: first/second; quality rankings require 100% success. HeteroMRTA is an architecture-level adaptation.

Setting	Method	SR (%) ↑	MS ↓	AWT ↓	FFT (s) ↓	RT (s) ↓
D15	MIP	97.5	44.73±17.28	4.59±2.73	26.05±20.86	322.67±11.23
	CP-SAT	**100.0**	**20.06±5.51**	**1.16±0.99**	40.01±37.98	254.40±85.37
	ALNS	**100.0**	26.61±5.15	2.24±1.21	0.96±0.31	310.51±3.97
	Greedy	**100.0**	42.44±9.30	10.53±6.18	**0.29±0.06**	**0.29±0.06**
	HeteroMRTA	**100.0**	39.11±6.77	6.84±2.29	1.04±0.25	3.34±0.26
	TAGS	**100.0**	36.49±5.53	5.80±2.21	1.90±0.36	5.19±0.41
D35	MIP	7.5	194.48±36.51	31.86±2.67	176.31±43.54	1085.06±86.28
	CP-SAT	35.0	92.09±50.65	9.15±6.53	363.60±195.82	920.50±6.04
	ALNS	**100.0**	**34.46±9.39**	**1.97±0.86**	10.57±3.15	928.16±16.04
	Greedy	**100.0**	40.48±7.89	4.11±2.39	**2.49±0.56**	**2.49±0.56**
	HeteroMRTA	**100.0**	37.10±5.23	2.91±0.86	2.87±0.42	7.71±0.68
	TAGS	**100.0**	35.16±4.40	2.10±0.82	5.87±0.37	13.48±0.70
D50	MIP	0.0	–	–	–	1686.28±268.85
	CP-SAT	2.5	238.92±0.00	29.97±0.00	1169.97±0.00	1264.90±17.47
	ALNS	**100.0**	46.29±15.05	2.47±1.09	33.27±9.99	1283.34±59.73
	Greedy	**100.0**	43.63±6.97	2.92±1.28	6.31±1.66	**6.31±1.66**
	HeteroMRTA	**100.0**	43.68±5.55	2.76±0.75	**4.92±0.71**	12.56±1.45
	TAGS	**100.0**	**40.62±4.35**	**1.80±0.54**	11.33±0.56	25.25±1.31

Metrics and computation. Success rate (SR) measures complete feasible schedules within the execution horizon. Makespan (MS), average collaborative waiting per robot (AWT), and first-feasible time (FFT, latency to the first complete schedule) use successful instances; runtime (RT) uses all instances. Learned methods and ALNS report instance-level means and population standard deviations after seed averaging. Appendix D.3 defines timing boundaries. All methods use eight cores of the same AMD Ryzen 9 9950X CPU per run.

4.2 Scheduling Quality and Runtime

To answer Q1, Table 2 compares D15, D35, and D50 with $M = N \in \{15, 35, 50\}$ and 50% collaboration. On D15, CP-SAT achieves MS 20.06, followed by ALNS at 26.61 and TAGS at 36.49; TAGS produces its best-of-16 result in 5.19 seconds. On D35, TAGS comes within 2.0% of ALNS's makespan (35.16 versus 34.46), at reported latencies of 13.48 and 928.16 seconds. On D50, TAGS has the lowest mean MS among full-completion methods: 40.62 versus 43.63 for the handcrafted Greedy baseline, a 6.9% reduction. Its AWT is 1.80 versus 2.47 for ALNS, the best external waiting-time baseline, a 27.1% reduction.

On D35 and D50, TAGS produces its first feasible schedule in 5.87 and 11.33 seconds, reducing FFT by 44.5% and 65.9% relative to ALNS. On D50, best-of-16 TAGS finishes in 25.25 seconds versus 1283.34 seconds for ALNS. MIP/CP-SAT solve 7.5%/35.0% of D35 and 0.0%/2.5% of D50 instances; quality averages use successful runs.

4.3 Robot Selection and Component Analysis

To answer Q2, Figure 3 compares TAGS with a uniform-robot-selection variant and three component ablations. Uniform selection samples an admissible robot, conditions features and context on it, and learns its command under the same training protocol. TAGS learns robot selection from the shared admissible-robot context. Uniform selection increases D50 mean MS by 3.5% and AWT by 25.0%, while reducing best-of-16 latency from 25.25 to 12.79 seconds. D50 cross-seed MS deviations are 0.81, 4.43, and 7.26 for TAGS, uniform selection, and HeteroMRTA, respectively (Appendix E).

Replacing the graph encoder with parameter-matched set attention raises MS and AWT by 13.6% and 27.8%. Removing attention-distance bias raises MS and AWT by 7.4% and 10.0%; command scoring retains robot-specific distances. The observed cross-seed MS standard deviation changes

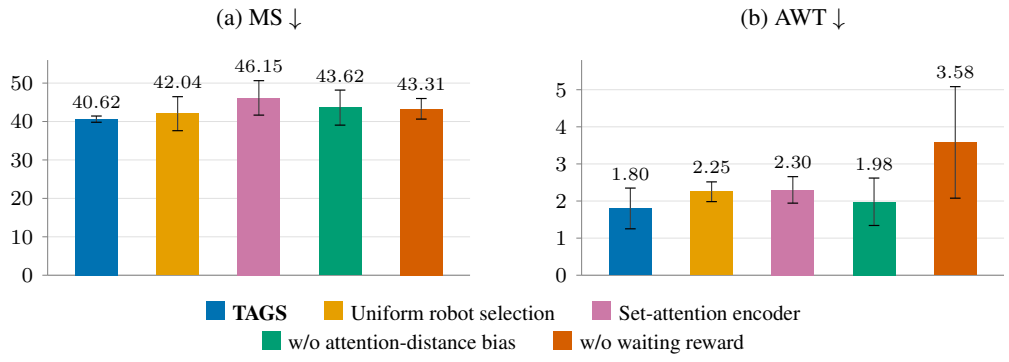


Figure 3: Ablations with 50 robots and 50 tasks (D50; best-of-16; 100% success). Bars show three-seed means; MS/AWT error bars are population standard deviations across seed means.

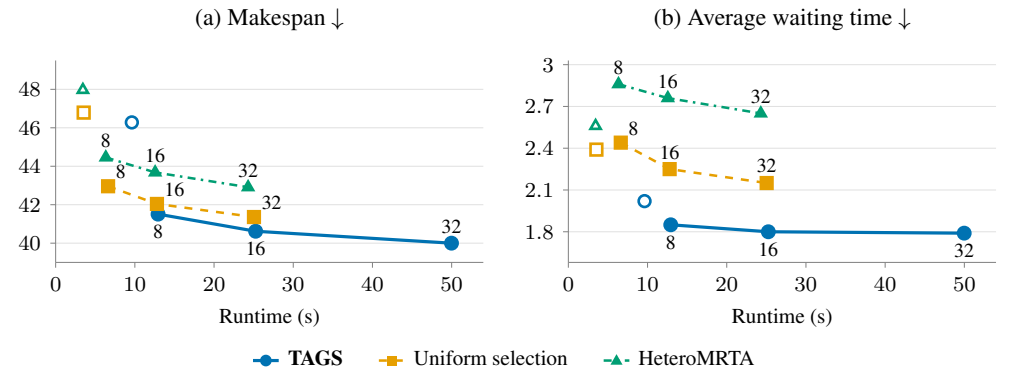


Figure 4: Three-seed decoding on D50 (50 robots and 50 tasks; 50% collaboration). Lines connect best-of-B budgets; hollow markers denote argmax. Runtime covers inference and simulation.

from 0.81 to 4.56. Removing the waiting reward increases AWT from 1.80 to 3.58 and MS from 40.62 to 43.31, connecting the auxiliary objective to reduced collaborative waiting.

4.4 Decoding Quality–Runtime Trade-Off

To answer Q3, Figure 4 evaluates fixed checkpoints with argmax and best-of-8, 16, and 32 decoding on D50. Best-of-B uses nested prefixes of one shared 32-candidate pool, choosing minimum makespan and using return to break exact ties. All individual rollouts finish every task. TAGS achieves the lowest mean MS and AWT among the learned methods at each candidate budget. At approximately 25 seconds, TAGS with 16 candidates attains MS 40.62 and AWT 1.80 versus 41.36 and 2.15 for uniform selection with 32 candidates, reducing MS by 1.8% and AWT by 16.3%.

Increasing TAGS from 16 to 32 candidates improves MS by 1.5% while nearly doubling latency, supporting the chosen main-experiment budget. Argmax TAGS obtains MS 46.28 in 9.62 seconds; Greedy obtains 43.63 in 6.31 seconds. The main D50 makespan advantage emerges with sampled candidate construction and selection.

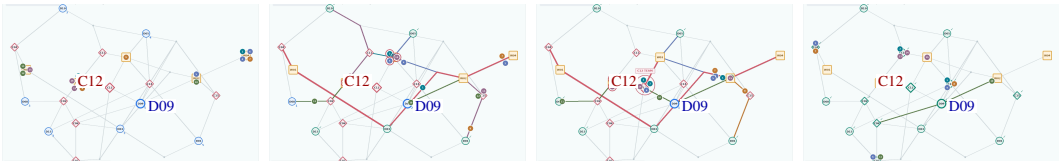
4.5 Transfer across Task Mixtures

Task composition changes how shared robots are allocated between cargo delivery and capability-covering teams. To answer Q4, we freeze the policy trained at 50% collaboration and evaluate it at 75% and 100% on paired D50 instances, keeping topology, robot attributes, and entity locations fixed (Table 3). TAGS completes every instance and achieves the lowest mean MS and AWT among full-completion methods at both ratios. Compared with HeteroMRTA, MS falls by 5.5% at 75% collaboration (41.93 versus 44.39) and 3.7% at 100% (43.00 versus 44.64).

Table 3: Task-mixture transfer with 50 robots and 50 tasks (D50; 40 paired instances): 50% collaboration is in-distribution (ID); 75%/100% are out-of-distribution (OOD). Learned entries average three seeds with best-of-16 decoding and frozen policies trained at 50%. The nominal 75% set contains 38 collaborative tasks.

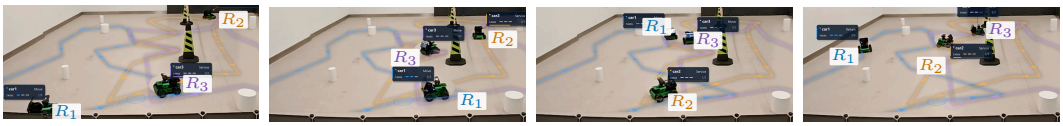
			50% (ID)			75% (OOD)			100% (OOD)		
Method	Budget (s)	Decode	SR (%) ↑	MS ↓	AWT ↓	SR (%) ↑	MS ↓	AWT ↓	SR (%) ↑	MS ↓	AWT ↓
MIP	1200	–	0	–	–	0	–	–	0	–	–
CP-SAT	1200	–	2.5	238.92	29.97	0	–	–	0	–	–
ALNS	60	–	**100**	61.45	2.88	32.5	75.78	4.27	0	–	–
ALNS	300	–	**100**	52.26	2.65	**100**	56.63	4.32	**100**	62.97	5.85
ALNS	1200	–	**100**	46.29	2.47	**100**	50.43	4.13	**100**	57.06	5.28
Greedy	–	Det.	**100**	43.63	2.92	**100**	49.88	4.65	**100**	53.52	4.86
HeteroMRTA	–	Bo16	**100**	43.68	2.76	**100**	44.39	4.39	**100**	44.64	5.71
TAGS	–	Bo16	**100**	**40.62**	**1.80**	**100**	**41.93**	**3.49**	**100**	**43.00**	**4.48**

(a) Simulation: 15 robots, 15 tasks



$t = 0$: Initialization | $t = 10.70$: 3 committed | $t = 12.79$: 4-robot team | $t = 25.47$: All tasks complete

(b) Physical execution: 3 robots



26.48 s | 39.73 s | 66.21 s | 92.69 s

Figure 5: Execution at different scales. (a) Fifteen robots serve mixed tasks; D09 and C12 highlight repeated delivery and team assembly (t: simulation time). (b) Three physical robots follow allocated routes; blue, orange, and purple identify R_1, R_2, and R_3. Bottom timestamps indicate video time.

ALNS reaches full completion at 300 seconds and improves quality by 1200 seconds; MIP/CP-SAT find no complete schedule at either transfer ratio. TAGS maintains full completion and its mean-quality ranking, supporting policy reuse across task mixtures.

Execution at simulation and physical scales. Figure 5(a) follows 15 robots serving 15 mixed tasks. Between $t = 10.70$ and 12.79, C12 progresses from three committed robots to a complete four-robot team executing synchronously, while D09's delivered quantity rises from three to four units. All tasks finish by $t = 25.47$; D09's five units are delivered in four capacity-feasible trips, illustrating repeated delivery alongside team assembly and execution. In Figure 5(b), TAGS allocates tasks online to three wheeled robots with software-configured heterogeneous capacities and capabilities in an arena with simple obstacles. Across the four frames, blue-labeled R_1 travels from the front-left to the central foreground, through the middle, and back along the left branch.

5 Conclusion

TAGS schedules heterogeneous robot teams performing repeated capacity-limited delivery and capability-dependent collaboration through an event-driven state representation and a topology-aware robot–command policy. Across the largest evaluated instances, it achieves the lowest mean makespan and collaborative waiting time among full-completion methods, and retains both rankings when the collaborative-task ratio increases without retraining. Future work will examine robot scarcity, larger task sets, and physical constraints coupling travel and collaboration.

AI USE STATEMENT

Generative AI tools assisted with manuscript organization and editing, notation refinement, interpretation of reported results, literature identification, and LaTeX source preparation.

REPRODUCIBILITY STATEMENT

Section 4.1 specifies instance generation, training, evaluation, and computation settings. Appendices A–D specify the simulator, policy, baselines, and evaluation protocol; Appendix E reports aggregate variability analyses and solver diagnostics. The benchmark generator, fixed evaluation instances, and implementation are provided in the anonymous supplementary material.

REFERENCES

Aakriti Agrawal, Amrit Singh Bedi, and Dinesh Manocha. RTAW: An attention inspired reinforcement learning method for multi-robot task allocation in warehouse environments. In *ICRA*, pp. 1393–1399, 2023.

Andrea Bendazzoli, Michele Urbani, Matteo Brunelli, and Francesco Pilati. A cooperative team orienteering optimisation model and a customised resolution metaheuristic. *Comput. Oper. Res.*, 163:106488, 2024.

Gerardo Berbeglia, Jean-François Cordeau, Irina Gribkovskaia, and Gilbert Laporte. Static pickup and delivery problems: A classification scheme and survey. *TOP*, 15(1):1–31, 2007.

Federico Berto, Chuanbo Hua, Laurin Luttmann, Jiwoo Son, Junyoung Park, Kyuree Ahn, Changhyun Kwon, Lin Xie, and Jinkyoo Park. PARCO: Parallel autoregressive models for multi-agent combinatorial optimization. In *NeurIPS*, volume 38, pp. 167130–167160, 2025.

Matteo Bettini, Ajay Shankar, and Amanda Prorok. Heterogeneous multi-robot reinforcement learning. In *AAMAS*, pp. 1485–1494, 2023.

Zhe Chen, Javier Alonso-Mora, Xiaoshan Bai, Daniel D. Harabor, and Peter J. Stuckey. Integrated task assignment and path planning for capacitated multi-agent pickup and delivery. *IEEE Robot. Autom. Lett.*, 6(3):5816–5823, 2021.

Han-Lim Choi, Luc Brunet, and Jonathan P. How. Consensus-based decentralized auctions for robust task allocation. *IEEE Trans. Robot.*, 25(4):912–926, 2009.

Weiheng Dai, Aditya Bidwai, and Guillaume Sartoretti. Dynamic coalition formation and routing for multirobot task allocation via reinforcement learning. In *IEEE International Conference on Robotics and Automation*, pp. 16567–16573, 2024.

Weiheng Dai, Utkarsh Rai, Jimmy Chiun, Yuhong Cao, and Guillaume Sartoretti. Heterogeneous multi-robot task allocation and scheduling via reinforcement learning. *IEEE Robot. Autom. Lett.*, 10(3):2654–2661, 2025.

M. Bernardine Dias, Robert Zlot, Nidhi Kalra, and Anthony Stentz. Market-based multirobot coordination: A survey and analysis. *Proc. IEEE*, 94(7):1257–1270, 2006.

Michael Drexl. Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints. *Transp. Sci.*, 46(3):297–316, 2012.

Véronique François, Yasemin Arda, Yves Crama, and Gilbert Laporte. Large neighborhood search for multi-trip vehicle routing. *Eur. J. Oper. Res.*, 255(2):422–441, 2016.

Brian P. Gerkey and Maja J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *Int. J. Robot. Res.*, 23(9):939–954, 2004.

Google OR-Tools. CP-SAT Solver. `https://developers.google.com/optimization/cp/cp_solver`, 2024. Accessed: 2026-06-18.

Philippe Grangier, Michel Gendreau, Fabien Lehuédé, and Louis-Martin Rousseau. An adaptive large neighborhood search for the two-echelon multiple-trip vehicle routing problem with satellite synchronization. *Eur. J. Oper. Res.*, 254(1):80–91, 2016.

M. H. Hà, T. D. Nguyen, T. Nguyen Duy, H. G. Pham, T. Do, and L.-M. Rousseau. A new constraint programming model and a linear programming-based adaptive large neighborhood search for the vehicle routing problem with synchronization constraints. *Comput. Oper. Res.*, 124:105085, 2020.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *ICLR*, 2019.

G. Ayorkor Korsah, Anthony Stentz, and M. Bernardine Dias. A comprehensive taxonomy for multi-robot task allocation. *Int. J. Robot. Res.*, 32(12):1495–1512, 2013.

Gilbert Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *Eur. J. Oper. Res.*, 59(3):345–358, 1992.

H. Li, H. Wang, J. Chen, and M. Bai. Two-echelon vehicle routing problem with time windows and mobile satellites. *Transp. Res. Part B Methodol.*, 138:179–201, 2020.

Jiazhen Liu, Glen Neville, Jinwoo Park, Sonia Chernova, and Harish Ravichandar. Learning and optimizing the efficacy of spatio-temporal task allocation under temporal and resource constraints, 2026. URL https://arxiv.org/abs/2601.02505.

S. Malik, M. Khonji, K. Elbassioni, and J. Dias. Collaborative last-mile delivery: A multi-platform vehicle routing problem with en-route charging, 2025.

Kunal Menda, Yi-Chun Chen, Justin Grana, James W. Bono, Brendan D. Tracey, Mykel J. Kochenderfer, and David H. Wolpert. Deep reinforcement learning for event-driven multi-agent decision processes. *IEEE Trans. Intell. Transp. Syst.*, 20(4):1259–1268, 2019.

Andrew Messing, Glen Neville, Sonia Chernova, Seth Hutchinson, and Harish Ravichandar. GRSTAPS: Graphically recursive simultaneous task allocation, planning, and scheduling. *The International Journal of Robotics Research*, 41(2):232–256, 2022.

Jinwoo Park, Harish Ravichandar, and Seth Hutchinson. Modeling and optimizing the provisioning of exhaustible capabilities for simultaneous task allocation and scheduling. In *AAMAS*, 2026. URL https://arxiv.org/abs/2602.13866.

Steve Paul, Payam Ghassemi, and Souma Chowdhury. Learning scalable policies over graphs for multi-robot task allocation using capsule attention networks. In *ICRA*, pp. 8815–8822, 2022.

Steve Paul, Wenyuan Li, Brian Smyth, Yuzhou Chen, Yulia R. Gel, and Souma Chowdhury. Efficient planning of multi-robot collective transport using graph reinforcement learning with higher order topological abstraction. In *ICRA*, pp. 5779–5785, 2023.

Steve Paul, Nathan Maurer, and Souma Chowdhury. Bigraph matching weighted with learnt incentive function for multi-robot task allocation. In *ICRA*, pp. 7250–7256, 2024.

Félix Quinton, Christophe Grand, and Charles Lesire. Market approaches to the multi-robot task allocation problem: a survey. *J. Intell. Robot. Syst.*, 107(2):29, 2023.

Yara Rizk, Mariette Awad, and Edward W. Tunstel. Cooperative heterogeneous multi-robot systems: A survey. *ACM Comput. Surv.*, 52(2):1–31, 2019.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *ICLR*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

Bruno Salezze Vieira, Glaydston Mattos Ribeiro, and Laura Bahiense. Metaheuristics with variable diversity control and neighborhood search for the heterogeneous site-dependent multi-depot multi-trip periodic vehicle routing problem. *Comput. Oper. Res.*, 153:106189, 2023.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *NeurIPS*, volume 28, pp. 2692–2700, 2015.

Zheyuan Wang, Chen Liu, and Matthew Gombolay. Heterogeneous graph attention networks for scalable multi-robot scheduling with temporospatial constraints. *Auton. Robot.*, 46(1):249–268, 2022.

Yuchen Xiao, Weihao Tan, and Christopher Amato. Asynchronous actor-critic for multi-agent reinforcement learning. In *NeurIPS*, volume 35, pp. 4385–4400, 2022.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In *NeurIPS*, volume 34, pp. 28877–28888, 2021.

Yu Zhang and Lynne E. Parker. IQ-ASyMTRe: Forming executable coalitions for tightly coupled multirobot tasks. *IEEE Trans. Robot.*, 29(2):400–416, 2013.

Zengyu Zou, Jingyuan Wang, Yixuan Huang, and Junjie Wu. Multi-agent pointer transformer: Seq-to-seq reinforcement learning for multi-vehicle dynamic pickup-delivery problems. *AAAI*, 40(19): 16593–16601, 2026.

A Extended Problem and Simulator Details

This appendix specifies the dispatch, completion, and event-update semantics underlying Section 3.2. It uses the main-text notation; $\leftarrow$ denotes an in-place update, and inequalities between capability vectors are componentwise.

A.1 Problem data and dynamic state

The connected undirected topology $\mathcal{G}$ has positive edge weights, and $d(u, v)$ is its shortest-path distance. Robots travel at a common speed $v > 0$, with $v = 1$ in the reported experiments. Task j has compatible storage set $\mathcal{W}_j$ and deterministic service duration $e_j \geq 0$; reported experiments set $e_j = 0$. Storage inventory is not depleted. Delivery quantities use $\{1, \ldots, q_{\max}\}$, with $q_{\max} = 3$.

At epoch n, t_n is physical time and $\mathcal{F}_n$ is the set of idle, unladen, uncommitted robots. Let $\rho_i \in \mathcal{W} \cup \{\perp\}$ identify robot i's current storage, with $\perp$ indicating that it is not at a storage node. An idle robot at a task node must move to storage before serving another task. During travel, p_i remains the last reached node; the destination and remaining event time retain travel progress. Traveling robots are excluded from the action candidates.

For a delivery task, $(m_j^{\mathrm{rem}}, m_j^{\mathrm{tr}})$ starts at $(m_j, 0)$. The amount available for a new dispatch is

$$m_j^{\mathrm{open}} = m_j^{\mathrm{rem}} - m_j^{\mathrm{tr}}. \tag{A.1}$$

For collaboration, the committed set $S_{j,n}$ and storage lock ω_j start empty and unset. The remaining requirement is $\mathbf{b}_j^{\mathrm{rem}} = [\mathbf{b}_j - \sum_{i \in S_{j,n}} \mathbf{c}_i]_+$. These quantities expose both undelivered cargo and capabilities already reserved by partial teams.

A.2 Task execution semantics

Storage movement. A storage command sends a free robot from p_i to p_w and schedules arrival at $t_n + d(p_i, p_w)/v$. At arrival, the robot's location and storage index are updated and it becomes idle. Movement does not reserve a task: command admissibility is recomputed from the arrival state.

Capacity-limited delivery. A free robot i at storage w can dispatch $\mathrm{D}(j, q)$ when

$$w \in \mathcal{W}_j, \qquad g_{i,k_j} = 1, \qquad 1 \leq q \leq \min\{q_{\max}, \kappa_i, m_j^{\mathrm{open}}\}. \tag{A.2}$$

Dispatch makes the robot busy with cargo (j, q) and updates $m_j^{\mathrm{tr}} \leftarrow m_j^{\mathrm{tr}} + q$. With zero loading time, its task-completion and release event occurs at $t_n + d(p_w, p_j)/v + e_j$. At that event,

$$m_j^{\mathrm{rem}} \leftarrow m_j^{\mathrm{rem}} - q, \qquad m_j^{\mathrm{tr}} \leftarrow m_j^{\mathrm{tr}} - q. \tag{A.3}$$

The robot's cargo is cleared and its location becomes p_j. The task completes when both quantities reach zero. Several robots can dispatch concurrently to the same task, including from different compatible storage nodes; each remains occupied until its own release event.

Capability-covering collaboration. A free robot i at storage w can join unfinished collaborative task j when

$$w \in \mathcal{W}_j, \quad \omega_j \in \{\perp, w\}, \quad i \notin S_{j,n}, \quad \mathbf{c}_i^\top \mathbf{b}_j^{\mathrm{rem}} > 0. \tag{A.4}$$

Each new member therefore supplies an unmet capability. The first commitment sets $\omega_j = w$; each commitment adds i to $S_{j,n}$ and makes that robot busy at storage with no finite scheduled event. Once $\mathbf{b}_j^{\mathrm{rem}} = \mathbf{0}$, the completed team departs automatically at $t_j^{\mathrm{dep}} = t_n$, without another policy action. All members share the completion and release time

$$C_j = t_j^{\mathrm{dep}} + d(p_{\omega_j}, p_j)/v + e_j. \tag{A.5}$$

For internal demand accounting, each member carries a share $1/|S_{j,n}|$ so that total in-transit and delivered amount equals one. This share is an accounting quantity; the collaboration command has no payload-quantity decision. A singleton team is allowed when one robot covers the full requirement. If robot i commits at $h_{i,j}$, its waiting contribution is $t_j^{\mathrm{dep}} - h_{i,j}$. Waiting ends at departure, and commitments persist until joint completion without withdrawal or reassignment.

Algorithm 1 Event-driven high-level scheduling episode

Require: Initial state s_0, masked policy π_θ, horizon $T_{\max}$
Initialize robot and task states; set $t \leftarrow 0$
loop
  if all tasks are complete and $t \leq T_{\max}$ **then**
    return successful schedule and makespan t
  else if $t \geq T_{\max}$ **then**
    return unsuccessful termination
  end if
  Construct $\mathcal{Y}(i)$ for every robot; set $\mathcal{I} \leftarrow \{i : \mathcal{Y}(i) \neq \varnothing\}$
  if $\mathcal{I} \neq \varnothing$ **then**
    Select (i, y) from the masked policy and apply its state update
  else if a busy robot has a finite scheduled event **then**
    Advance t to the earliest event; process one event and update the state
  else
    return failure (no admissible continuation)
  end if
end loop

A.3 Work-conserving admissible actions

For a free robot hypothetically placed at storage w, let $\mathcal{U}_n(i, w)$ be the union of delivery and collaborative commands admitted by Equations (A.2) and (A.4). Finished tasks and delivery tasks with no open demand contribute no command. The storage nodes offering service are

$$\mathcal{H}_n(i) = \{w \in \mathcal{W} \setminus \{\rho_i\} : \mathcal{U}_n(i, w) \neq \varnothing\}. \tag{A.6}$$

The reported configuration uses the conditional command set

$$\mathcal{Y}_n(i) = \begin{cases} \mathcal{U}_n(i, \rho_i), & i \in \mathcal{F}_n,\ \rho_i \neq \perp,\ \mathcal{U}_n(i, \rho_i) \neq \varnothing, \\ \{\mathrm{W}(w) : w \in \mathcal{H}_n(i)\}, & i \in \mathcal{F}_n \text{ otherwise}, \\ \varnothing, & i \notin \mathcal{F}_n. \end{cases} \tag{A.7}$$

Thus a robot with local service chooses a service command; otherwise it may move to storage that currently enables service. The action vocabulary has no voluntary-wait command. This work-conserving construction restricts the policy's schedule search by removing local delays and repositioning while service is available.

The robot and command masks are the indicators of $\mathcal{Y}_n(i) \neq \varnothing$ and $y \in \mathcal{Y}_n(i)$, respectively. Writing $\mathcal{I}_n = \{i : \mathcal{Y}_n(i) \neq \varnothing\}$ gives

$$\mathcal{A}_n = \{(i, y) : i \in \mathcal{I}_n,\ y \in \mathcal{Y}_n(i)\}, \qquad \operatorname{supp} \pi_\theta(\cdot \mid s_n) \subseteq \mathcal{A}_n. \tag{A.8}$$

The simulator rechecks the selected command before execution. These masks enforce current command conditions; the availability of a complete continuation is determined by subsequent team commitments and events.

A.4 Event advancement and termination

Algorithm 1 gives the policy-independent episode loop. Consecutive commands can dispatch several robots at the same physical time. When no robot has a command, the simulator processes the earliest finite event and rechecks the state. Tied events require no additional time advancement and can be interleaved with scheduling decisions at that timestamp. Success requires completing every task by $T_{\max}$; otherwise reaching the horizon ends the episode unsuccessfully. A state with no admissible command and no finite event also terminates unsuccessfully. In particular, members of incomplete teams have infinite event times, and partial commitments can leave no remaining combination of robots that completes a team. The generator ensures that every requirement is initially coverable.

A.5 Reward and execution model

Equation (8) uses W_n, the accumulated collaborative waiting through epoch n, including elapsed portions of ongoing waits. The reported training configuration disables progress and task-

Table A1: Zero-based layout of the 39-dimensional entity state. Unlisted entries within each index range are reserved and remain zero.

Indices	Task	Storage	Robot
0–2	Task-type one-hot vector	Zero	Delivery-type compatibility vector
3	Completion flag	Fraction of idle robots at storage	Idle flag
4	In-transit assignment or satisfied collaboration	Zero	Travel-progress scalar
5–6	Incoming-robot fraction and service time	Zero	Zero
7	Entity-type code 0	Entity-type code 3	Entity-type code 2
8–11	Active-task role fields	Storage indicator	Away/at-storage and loaded-state flags
12–14	Zero	Types of compatible open tasks	Zero
15–18	Collaboration flag and team progress	Available collaboration and maximum progress	Collaboration phase and current wait
19–28	Remaining and original capability requirements	Aggregated outstanding capability requirements	Zero
29–33	Zero	Zero	Collaborative-capability vector
34–38	Remaining, original, and in-transit demand	Zero	Payload capacity and loaded quantity

completion shaping terms. With $\gamma = 1$ and $\beta = 0.1$, a successful episode has return $-C_{\max} - \beta W_{\text{final}}$; unsuccessful termination uses terminal penalty $-T_{\max}$.

Travel times come from static shortest paths and a common robot speed. Loading and service durations are zero in the experiments, and inventory is unlimited. The high-level model represents capacity, capability coverage, storage compatibility, and synchronization. Low-level collision avoidance, congestion, and motion uncertainty are handled separately from these scheduling transitions. When physical arrival times differ from planned times, execution updates must reach the scheduler or trigger replanning.

B Policy Architecture and Optimization

This appendix specifies the tensors, context construction, and optimization used by TAGS and its uniform-robot-selection ablation. Both use the same entity features, topology-aware encoder, pointer readout, waiting reward, and critic design.

B.1 Observation and entity features

Padded entity tensor. Valid entities are ordered as tasks, storage locations, and robots and padded to 50, 5, and 50 entries, respectively. Each entity has a 39-dimensional type-marked feature vector (Table A1). Fields inapplicable to an entity type remain zero. Task demand is normalized by 6, robot capacity and load by 3, robot and incoming counts by 50, and collaborative waiting by 200. Binary fields use $\{0, 1\}$. A padded entity has -1 in its first coordinate and is excluded from attention keys, pooling, and readout.

Distances and reference conditioning. The complete entity graph receives unnormalized shortest-path travel times d_{uv} between current entity locations. Task and storage locations are fixed; a robot's incident distances update when it reaches a new node. Task–storage compatibility and command admissibility enter through masks. For uniform selection, the reference-distance feature is measured from the selected robot. For TAGS, it is the mean distance from the admissible robots:

$$\bar{d}_u = \begin{cases} d_{iu}, & \text{uniform selection, selected robot } i, \\ |\mathcal{I}_n|^{-1} \sum_{i \in \mathcal{I}_n} d_{iu}, & \text{TAGS}. \end{cases} \tag{B.1}$$

Appending this scalar gives the 40-dimensional encoder input.

B.2 TOPOLOGY-AWARE ENCODER

Inputs project to width 256 and pass through four dense graph-attention layers with eight heads of width 32 (Veličković et al., 2018; Ying et al., 2021). Equation (10) gives the attention operation. For its single edge channel, the implementation uses a shared scalar affine projection, layer-specific affine projections, and head-specific mixing. Consequently, each head's travel-time bias is affine in d_{uv}. The head-mixing coefficient is initialized to zero, so attention initially has no distance-logit perturbation. Padding keys are masked before normalization.

Each layer applies value aggregation, output projection, residual LayerNorm, and a gated feed-forward residual:

$$\begin{aligned} \widetilde{\mathbf{h}}_u &= \mathrm{LN}\left(\mathbf{h}_u + W_o \sum_v \alpha_{uv} W_v \mathbf{h}_v\right), \\ \mathbf{h}'_u &= \widetilde{\mathbf{h}}_u + W_2\left[\mathrm{sigmoid}(W_g \bar{\mathbf{h}}_u) \odot W_f \bar{\mathbf{h}}_u\right], \qquad \bar{\mathbf{h}}_u = \mathrm{LN}(\widetilde{\mathbf{h}}_u), \end{aligned} \tag{B.2}$$

where the learned projections W_g, W_f expand to width 512, W_2 returns to width 256, and $\odot$ denotes elementwise multiplication. For $V = M + N + |\mathcal{W}|$ valid entities and embedding width d, dense attention costs $O(V^2 d)$ per layer and stores $O(V^2)$ attention and distance entries.

B.3 REFERENCE-CONDITIONED CONTEXT AND READOUT

Let $\bar{\mathbf{h}}_{\mathcal{T}}$ and $\bar{\mathbf{h}}_{\mathcal{E}}$ be padding-masked means over task and all entity embeddings. For robot summary $\mathbf{r}$, the shared context construction is

$$\mathcal{C}(\mathbf{r}) = \mathrm{Dec}_2\big(W_c[\mathbf{r} \| \bar{\mathbf{h}}_{\mathcal{T}} \| \bar{\mathbf{h}}_{\mathcal{E}}], H_{\mathcal{T}}\big), \tag{B.3}$$

where $\|$ concatenates vectors, W_c projects to 256 dimensions, and Dec_2 consists of two cross-attention layers with task embeddings $H_{\mathcal{T}}$ as memory. The memory mask removes padding but retains completed tasks and their states. TAGS uses

$$\bar{\mathbf{z}} = \mathcal{C}\left(|\mathcal{I}_n|^{-1} \sum_{i \in \mathcal{I}_n} \mathbf{h}_i\right), \qquad \mathbf{z}_i = \mathcal{C}(\mathbf{h}_i), \tag{B.4}$$

where $\bar{\mathbf{z}}$ scores admissible robots and $\mathbf{z}_i$ scores robot-conditioned commands. All robot-conditioned contexts and command logits are computed in one forward pass. Uniform selection uses $\mathbf{z}_i$ for its externally sampled robot.

The normalized, clipped pointer (Vinyals et al., 2015) is

$$\mathrm{ptr}(\mathbf{z}, \mathbf{h}) = 10 \tanh\left(\frac{(W_q \,\mathrm{LN}(\mathbf{z}))^\top (W_k \,\mathrm{LN}(\mathbf{h}))}{\sqrt{256}}\right), \tag{B.5}$$

where W_q, W_k are learned query and key projections. Delivery logits follow Equation (11); storage logits are $e_{i,w} = \mathrm{ptr}(\mathbf{z}_i, \mathbf{h}_w) + \lambda_d d_{iw}$. The learned, sign-unconstrained scalar λ_d is shared by task and storage readouts within an actor. Collaborative tasks use the task readout with only the first quantity slot unmasked. Environment masks are applied after logits are assembled.

B.4 ROBOT-SELECTION DISTRIBUTIONS

Uniform selection draws i uniformly from $\mathcal{I}_n$ outside the learned policy and optimizes only the conditional command log-probability. TAGS learns both factors in Equation (9), with robot logits $\mathrm{ptr}(\bar{\mathbf{z}}, \mathbf{h}_i)$. Stochastic decoding samples a robot and then its command. Argmax decoding maximizes the joint probability:

$$(i_n^*, y_n^*) = \arg \max_{(i,y) \in \mathcal{A}_n} \left[\log \pi_\theta(i \mid s_n) + \log \pi_\theta(y \mid s_n, i)\right]. \tag{B.6}$$

The padded command layout contains five storage slots and 50×3 task–quantity slots, giving 155 conditional entries per robot. The learning interface stores the selected pair as one categorical action and executes it as one simulator transition. This factorization applies to one event decision.

Table A2: Optimization configuration shared by the learned methods.

Setting	Value
Optimizer	Adam, moment coefficients $(0.9, 0.999)$, $\epsilon_{\mathrm{Adam}} = 10^{-6}$
Initial learning rate	10^{-4}
Learning-rate schedule	Cosine decay to 0.05 of initial rate
Discount / GAE	$\gamma = 1$, $\lambda_{\mathrm{GAE}} = 0.99$
PPO clip / entropy weight	0.3 / 0
Critic loss / coefficient	Squared error / 1
Waiting-reward weight	$\beta = 0.1$
Gradient-norm limit	5
Parallel environments	4
Transitions per update	1500
Minibatch / PPO epochs	750 / 4
Training length	160,500 transitions
Episode limits	2000 decisions, $T_{\max} = 500$

B.5 Critic and PPO optimization

An independent critic repeats the graph encoder and context construction, then uses a $256 \to 256 \to 1$ ReLU MLP to predict $V_\phi(s_n)$. Its parameters ϕ are separate from the actor's, and it has no pointer-distance coefficient. The centralized scheduler uses standard clipped PPO (Schulman et al., 2017), implemented through MAPPO infrastructure. With the importance ratio $\varrho_n = \exp[\log \pi_\theta(a_n \mid s_n) - \log \pi_{\theta_{\mathrm{old}}}(a_n \mid s_n)]$ and GAE estimate $\widehat{A}_n$, the actor maximizes

$$\mathbb{E}_n\Big[\min\{\varrho_n \widehat{A}_n,\ \mathrm{clip}(\varrho_n, 1-\epsilon, 1+\epsilon)\widehat{A}_n\}\Big]. \tag{B.7}$$

For TAGS, the joint log-probability is the sum of the two factors; for uniform selection it contains only the command factor.

GAE (Schulman et al., 2016) is computed backward as

$$\begin{aligned} \Delta_n &= r_n + \gamma(1 - z_n^{\mathrm{term}})V_\phi(s_{n+1}) - V_\phi(s_n), \\ \widehat{A}_n &= \Delta_n + \gamma\lambda_{\mathrm{GAE}}(1 - z_n^{\mathrm{done}})\widehat{A}_{n+1}, \\ \widehat{V}_n &= \widehat{A}_n + V_\phi(s_n), \end{aligned} \tag{B.8}$$

where z_n^{term} marks a true terminal state and z_n^{done} also marks truncation. Advantages are not normalized. Advantages and value targets are computed once per collected batch and reused for four PPO epochs. The critic minimizes unclipped squared value error; value clipping and entropy regularization are disabled.

Actor and critic use separate Adam optimizers (Table A2). Each collection batch aggregates 1500 transitions from four environments. Two 750-transition minibatches are reused for four epochs, giving eight actor and eight critic updates per batch. For collected transition count f and nominal budget $F = 160{,}000$, both learning rates follow

$$\eta(f) = 10^{-4}\left[0.05 + 0.95\,\frac{1 + \cos(\pi \min\{f/F, 1\})}{2}\right]. \tag{B.9}$$

Collecting complete batches yields 107 batches and a final checkpoint at 160,500 transitions. Training seeds are 20260706, 20260707, and 20260708. Instances are generated online from the distribution in Section 4.1; training-time evaluation is disabled. All reported evaluations use the final checkpoint without validation- or test-based selection. TAGS uses activation checkpointing in actor and critic encoders to recompute attention activations during backpropagation while preserving the forward computation and objective.

C Baseline Implementation Details

MIP, CP-SAT, and ALNS optimize complete routes offline for each instance. Greedy and the learned policies issue robot–command decisions from the current event state. We evaluate each method as a

standalone scheduler using its native initialization. MIP and CP-SAT receive no externally supplied initial solution; ALNS constructs its initial solution through BestInsertion. Complete schedules from Greedy or the learned policies can also provide candidate initial solutions for subsequent search. Every method receives eight CPU cores on the same processor.

C.1 Common semantics and validation

All methods read the same topology distances, initial positions, capacities, delivery compatibilities, collaborative capabilities, demands, and compatible-storage sets. A delivery stop (j, w, q) assigns quantity q of task j from storage w to one robot. A task can appear in multiple stops and routes, whose quantities must sum to its demand. A collaborative stop specifies one task, shared storage, and capability-covering team; each member has a corresponding synchronized route event.

Search methods minimize full-completion makespan. ALNS uses an incomplete-solution penalty during repair and zero waiting-time tie-break weight. Deterministic replay checks travel, capacity, compatibility, demand, capability coverage, and synchronization against Appendix A. Success requires a complete legal replay within the execution horizon, and quality metrics include only successful instances.

C.2 Slot-indexed MIP and CP-SAT

The two baselines use equivalent slot-indexed formulations adapted from synchronized routing models (Malik et al., 2025; Hà et al., 2020; Bendazzoli et al., 2024). Let $p \in \{1, \dots, L\}$ index ordered activity slots on each robot route. An occupied slot is one delivery trip or one participation in a collaborative task. Variables are instantiated for compatible robots, allowed storage nodes, and finite travel legs.

Delivery assignment and slot use. Binary x_{ipjw} assigns delivery task j from storage w to robot i in slot p, with integer quantity q_{ipjw}. Binary y_{ipjw} assigns collaborative participation; a_{ip} indicates an occupied slot. Their constraints include

$$\begin{aligned} a_{ip} &= \sum_{j \in \mathcal{T}^D} \sum_{w \in \mathcal{W}_j} x_{ipjw} + \sum_{j \in \mathcal{T}^C} \sum_{w \in \mathcal{W}_j} y_{ipjw}, \\ x_{ipjw} &\le q_{ipjw} \le \kappa_i x_{ipjw}, \\ \sum_{i \in \mathcal{R}} \sum_{p=1}^{L} \sum_{w \in \mathcal{W}_j} q_{ipjw} &= m_j, \qquad j \in \mathcal{T}^D. \end{aligned} \tag{C.1}$$

A delivery task can therefore use several slots, robots, and compatible storage nodes.

Team and storage selection. Let $\mathcal{K}_j = \{K_{j1}, \dots, K_{jH_j}\}$ contain all inclusion-minimal robot teams covering $\mathbf{b}_j$. Binary u_{jk} selects team K_{jk}, v_{jw} selects storage, and z_{ij} denotes membership:

$$\begin{aligned} \sum_{k=1}^{H_j} u_{jk} = 1, \qquad \sum_{w \in \mathcal{W}_j} v_{jw} = 1, \qquad z_{ij} = \sum_{k : i \in K_{jk}} u_{jk}, \\ \sum_{p=1}^{L} \sum_{w \in \mathcal{W}_j} y_{ipjw} = z_{ij}, \qquad y_{ipjw} \le z_{ij}, \quad y_{ipjw} \le v_{jw}. \end{aligned} \tag{C.2}$$

Every selected member has exactly one corresponding activity, and all members use the same storage.

Route order and synchronization. Let F_{ip} be robot i's finish time after slot p, t_j^{dep} the shared departure time, and H a valid horizon. Used slots form a prefix:

$$\begin{aligned} F_{i0} = 0, \qquad a_{ip} \ge a_{i,p+1} \quad (p < L), \\ F_{i,p-1} \le F_{ip} \le F_{i,p-1} + H a_{ip}, \qquad C_{\max} \ge F_{iL}. \end{aligned} \tag{C.3}$$

Write an activity as $r = (\tau, j, w)$, where τ is its delivery/collaboration type. Its binary assignment A_{ipr} is the corresponding x_{ipjw} or y_{ipjw}; its endpoint is $\nu(r) = p_j$ and service-leg duration is

Table A3: Backend differences for the shared slot-indexed formulation. All runs receive eight CPU cores.

Item	MIP	CP-SAT
Backend	OR-Tools interface, SCIP 10.0.0	OR-Tools CP-SAT
Time domain	Continuous	Integer, scale 1000
Conditional timing	Big-M inequalities	Enforced constraints
First incumbent	Not exposed by solver interface	Solver callback

$\ell_{jw} = d(p_w, p_j)/v + e_j$. Slot zero is a fixed dummy activity at the robot's initial node. For predecessor r' and current delivery,

$$F_{ip} \geq F_{i,p-1} + \frac{d(\nu(r'), p_w)}{v} + \ell_{jw} - B_{\text{big}}(2 - A_{i,p-1,r'} - x_{ipjw}), \tag{C.4}$$

where B_{big} is a valid Big-M constant. Collaborative participation instead imposes

$$\begin{aligned} t_j^{\text{dep}} &\geq F_{i,p-1} + \frac{d(\nu(r'), p_w)}{v} - B_{\text{big}}(2 - A_{i,p-1,r'} - y_{ipjw}), \\ F_{ip} &\geq t_j^{\text{dep}} + \ell_{jw} - B_{\text{big}}(1 - y_{ipjw}). \end{aligned} \tag{C.5}$$

The common departure occurs after all selected members reach storage. CP-SAT uses conditional enforcement for these timing constraints.

Objective and configuration. Both models minimize $C_{\max}$ and impose the per-robot lower bound

$$C_{\max} \geq \sum_{p=1}^{L} \sum_{r} \ell_{j(r),w(r)} A_{ipr}, \qquad i \in \mathcal{R}. \tag{C.6}$$

This bound sums selected service legs and omits nonnegative inter-activity travel. The experiments set $L = 8$, allowing at most eight service activities per robot. This is a search-model setting; the simulator's execution horizon is $T_{\max} = 500$. Runs with $L = 10$ and $L = 12$ under the same time limits gave no improvement, so the main results retain $L = 8$. Both solvers start cold with seed 20260609 and receive 300, 900, and 1200 seconds on D15, D35, and D50. The solver time limit excludes model construction; reported RT includes construction, extraction, and replay. Table A3 gives backend details.

C.3 Adaptive large neighborhood search

ALNS adapts synchronized coalition insertion and neighborhood search from cooperative team orienteering (Bendazzoli et al., 2024). Routes retain integer quantities for each delivery stop. A collaborative event is inserted into all selected members' routes with common task and storage identifiers. Replay enforces the shared execution semantics.

BestInsertion. Starting from empty or partially destroyed routes, delivery candidates enumerate quantities, compatible storage, and route positions. Their priority is q/m_j, while a complete collaborative insertion has priority one. Candidates are ranked by priority divided by incremental route cost. The reported configuration validates at most the top 32 delivery candidates per insertion step. For collaboration, it chooses an anchor, ranks capability-covering teams by member insertion costs, checks at most 24 teams, and retains the feasible synchronized insertion with the lowest waiting time.

Let $J(Q)$ be the replayed makespan of complete route set Q. During repair, each missing task and an invalid partial schedule add a penalty of 10^6; the waiting-time tie-break weight is zero. Algorithm 2 records the search loop.

Neighborhoods and adaptation. The destroy pool contains random, worst, length-2/3/4 sequence, relatedness, and largest-wait removal. Local search includes exchange, within-route reordering, replacement, two-route tail exchange, and cross-route relocation. A collaborative event moves or is removed as one synchronized group. After three iterations without a new best, the removal range doubles from 10–20% to 20–40% of route events. New global best, previously unseen

Algorithm 2 ALNS for joint delivery and collaborative scheduling

Require: Instance, time limit, iteration limit
$Q \leftarrow \mathrm{BestInsertion}(\varnothing)$; $Q^* \leftarrow Q$; initialize all operator weights to 1
Initialize temperatures T and $T_{\min}$ from $J(Q)$
while $T \geq T_{\min}$ and budgets remain **do**
    Set removal range from route-event count and stagnation
    Sample destroy operator h_D and local-search operator h_L by roulette
    $\widetilde{Q} \leftarrow h_D(Q)$; $\widetilde{Q} \leftarrow h_L(\widetilde{Q})$
    $\widetilde{Q} \leftarrow \mathrm{BestInsertion}(\widetilde{Q})$; draw $u \sim U(0,1)$
    if $J(\widetilde{Q}) \leq J(Q)$ or $u \leq \exp((J(Q) - J(\widetilde{Q}))/T)$ **then**
        $Q \leftarrow \widetilde{Q}$
        if $J(Q) < J(Q^*)$ **then**
            $Q^* \leftarrow Q$; reapply h_L and BestInsertion while improving Q^*
        end if
        Accumulate the outcome score for (h_D, h_L)
    end if
    $T \leftarrow 0.999T$; update weights every 20 iterations and reset every 100
end while
Deterministically replay and **return** Q^*

accepted nonworsening, and accepted worsening solutions score 35, 12, and 5, respectively. Every 20 iterations, operator h, used $n_h > 0$ times with accumulated score S_h, updates its weight as

$$\omega_h \leftarrow (1-\zeta)\omega_h + \zeta S_h/n_h, \qquad \zeta = 0.1. \tag{C.7}$$

Unused weights are multiplied by $1-\zeta$, and all weights reset to one every 100 iterations. Initial and terminal temperatures are $0.1|J(Q)|/[-\log(0.5)]$ and $0.005|J(Q)|/[-\log(0.5)]$. The maximum is 100,000 iterations. Relatedness coefficients are $(3, 13, 7)$; the middle time-window term is zero because this benchmark has no time windows. Time limits are 300, 900, and 1200 seconds, and main results average three independent search seeds. The 60- and 300-second transfer entries are incumbent snapshots of the corresponding 1200-second trajectories.

C.4 Greedy and adapted HeteroMRTA

Greedy. Greedy is a handcrafted heuristic in our event-driven framework, using the same robot–command interface and masks as the learned policies. At storage, it orders delivery commands by descending quantity and then ascending storage-to-task distance. A storage move is ranked by the largest downstream feasible quantity and then the combined robot-to-storage and storage-to-task distance. Robot selection compares each robot's best command, capacity, and distance lexicographically. Collaboration uses the capability-progress condition in Equation (A.4). Greedy generates one rollout, so FFT equals runtime for successful rollouts.

HeteroMRTA. The learning baseline adapts the AttentionNet architecture of Dai et al. (2025) to the shared joint-scheduling environment. Tasks and storage form the site set, with robots in a second set. Separate one-layer, eight-head self-attention encoders are followed by one layer of bidirectional cross-attention, all at width 256. This depth approximately matches the parameter count of TAGS's four-layer encoder. The adaptation shares the 39-dimensional states, reference-distance feature, command space, masks, pointer readout and its distance term, critic design, PPO configuration, and reward. It uses uniform robot selection. Its encoder processes robot/site features without the full pairwise distance tensor or attention-distance bias. Training seeds, final-checkpoint selection, and best-of-B evaluation match the other learned methods.

D Benchmark Construction and Evaluation Protocol

D.1 Training distribution and held-out sets

Instances are generated online from Table 1. The task count is uniform on the integers 15–50. Five robot species are used, with 3–10 robots per species and the total capped at 50. Storage count is

Table A4: Fixed held-out evaluation sets. Collaborative-task count is $\mathrm{round}(0.5M)$.

Set	Tasks M	Robots N	Collaborative tasks	Instances
D15	15	15	8	40
D35	35	35	18	40
D50	50	50	25	40

uniform on 2–5. The collaborative count is $\mathrm{round}(0.5M)$. Delivery type is uniform over three types, demand over 2–6, and capacity over 1–3. All robots support the three delivery types. Their five-dimensional collaborative capabilities cycle through ten fixed two-hot patterns, and every collaborative requirement is a nonzero binary vector coverable by the generated team. For each task, a compatible-storage count is sampled uniformly from $1, \ldots, |\mathcal{W}|$, followed by a uniformly sampled subset of that size.

Each topology has $M + |\mathcal{W}| + 5$ nodes. Generation first constructs a random spanning tree and then augments edges toward sampled degrees in $\{2, 3, 4\}$. Edge weights are uniform on $[1, 5]$. Tasks and storage occupy distinct nodes; robots start at uniformly sampled storage. All-pairs shortest paths define unit-speed travel times. Reported instances have zero loading and service durations and nondepleting inventory.

The fixed test sets were generated once with seed 20260704 and serialized independently of training. Table A4 gives their composition. These instances are used only for reporting final-checkpoint performance.

D.2 Paired collaboration-ratio transfer sets

The 50%, 75%, and 100% D50 sets are paired instance families, with nested collaborative-task subsets. They contain 25, 38, and 50 collaborative tasks, respectively; the nominal 75% setting is 76% after integer rounding. Topology, robots, initial positions, capacities, capabilities, storage, task positions and types, and compatible-storage sets remain fixed within a family. Converting a delivery task changes its collaboration flag, sets its demand to one, and assigns a coverable nonzero binary capability requirement. Requirements introduced at 75% are retained at 100%. Conversion order and requirements use derivation seed 20260720. This construction varies task composition while preserving the underlying topology and entity placement.

D.3 Nested candidate decoding and timing

Main learned results use best-of-16 stochastic rollouts. Candidate seeds 0–31 define one pool for each checkpoint and instance; budgets 8, 16, and 32 use nested prefixes 0–7, 0–15, and 0–31. D15, D35, and transfer sets use 16 candidates, while the D50 budget study also evaluates 8 and 32. Argmax is evaluated separately. Each evaluation rollout permits 1500 policy decisions and simulated horizon $T_{\max} = 500$.

All methods are evaluated on the same AMD Ryzen 9 9950X CPU with eight CPU cores allocated to each run. Candidate generation is CPU-only, with policies preloaded and batch size one. The decoder assigns candidates to eight persistent workers, one candidate per worker per round; 16 candidates therefore take two rounds. Among completed candidates it selects minimum makespan, breaking exact ties by return. If no candidate completes, the instance is unsuccessful; partial-progress and return criteria serve only diagnostic output.

Let $t_{k,c}$ be the inference-and-simulation duration of candidate c on instance k. Candidate c is assigned to worker $c \bmod 8$, and reported parallel decoding time is

$$t_k^{\mathrm{run}}(B) = \max_{w \in \{0,\ldots,7\}} \sum_{\substack{0 \le c < B \\ c \bmod 8 = w}} t_{k,c}. \tag{D.1}$$

FFT is the earliest cumulative worker time at which a successful candidate completes. These timers include policy inference and simulator stepping with preloaded policies. Checkpoint loading, worker launch and staggering, environment reset, metric extraction, aggregation, and serialization are out-

Table A5: Attention-distance bias across scales. Entries are mean ± population standard deviation across three seed-level means, each evaluated on 40 instances. All runs achieve 100% SR.

Set	Variant	MS	AWT
D15	TAGS	36.490 ± 1.717	5.804 ± 1.138
	w/o attention-distance bias	38.853 ± 3.755	6.257 ± 0.669
D35	TAGS	35.163 ± 0.560	2.103 ± 0.472
	w/o attention-distance bias	37.302 ± 4.401	2.418 ± 0.440
D50	TAGS	40.619 ± 0.810	1.803 ± 0.549
	w/o attention-distance bias	43.618 ± 4.558	1.976 ± 0.639

side the timer. MIP/CP-SAT RT includes model construction, extraction, and deterministic replay in addition to solver time.

Candidate-level success is measured before selection. Across the three D50 checkpoints of each learned method, all $3 \times 40 \times 32$ sampled candidates complete all tasks.

D.4 Success and statistical aggregation

A schedule succeeds when replay satisfies every modeled constraint and completes all tasks by $T_{\max}$. SR averages this binary outcome. MS, AWT, and FFT use successful schedules; RT includes all instances. AWT sums collaborative waiting across robots and divides by N. Under best-of-B, one successful candidate suffices for instance-level success.

The learned methods use training seeds 20260706–20260708, and main ALNS results use three independent search seeds. Measurements are first averaged over seeds for each fixed instance and then summarized over the 40 instances. Failed runs do not enter success-conditioned metrics. The main-table deviations are population standard deviations of these instance-level averages. Greedy and the exact solvers use one run per instance. The seed analyses in Appendix E instead summarize three seed-level means, each obtained on the same 40 instances. These two deviations describe instance variability and training-seed variability, respectively.

E Additional Analyses and Solver Diagnostics

This appendix summarizes variation across training seeds, extends the distance-bias analysis across scales, and records exact-solver diagnostics. All learned entries use the same fixed instances and best-of-16 decoding.

E.1 Aggregate variation across training seeds

TAGS has the lowest three-seed mean MS and AWT among the learned methods at each size. Its cross-seed MS standard deviations on D15, D35, and D50 are 1.72, 0.56, and 0.81, compared with 3.76, 3.64, and 4.43 for uniform selection and 4.44, 5.28, and 7.26 for HeteroMRTA. These population deviations summarize three seed-level means, each evaluated on the same 40 fixed instances.

E.2 Attention-distance bias across scales

Removing the encoder's attention-distance bias increases mean MS by 6.5%, 6.1%, and 7.4% on D15, D35, and D50. Using the more precise seed aggregates in Table A5, AWT increases by 7.8%, 15.0%, and 9.6%. The reduction in cross-seed MS variation is especially pronounced on D35 and D50. The robot-specific distance term in command readout remains enabled in this ablation.

E.3 Exact-solver diagnostics

CP-SAT finds complete incumbents on 40, 14, and 1 of the 40 D15, D35, and D50 instances; MIP finds 39, 3, and 0. Thirteen CP-SAT D15 instances are proven optimal. Unsuccessful runs terminate without a complete incumbent. Table A6 reports incumbent-to-bound gaps and slot usage for the

Table A6: Eight-slot solver diagnostics. Gap is the mean relative incumbent-to-bound gap over replay-valid incumbents; the displayed deviation is a population standard deviation. Slots is the maximum used by any robot. Cap hits count valid incumbents with at least one robot using all eight slots.

Method	Set	Gap (%)	Slots	Cap hits
CP-SAT	D15	31.0	5	0/40
	D35	77.3	8	9/14
	D50	95.8	8	1/1
MIP	D15	75.98 ± 8.31	7	0/39
	D35	94.9	8	3/3
	D50	–	–	–

eight-slot formulation. Some D35 and D50 incumbents use every slot; increasing the cap to 10 or 12 enlarged the search without improving the results under the same time limits.

F Reproducibility and Execution Settings

F.1 Artifacts and randomness control

The anonymous artifact contains the benchmark generator, fixed JSON instances, transfer-set derivation and validation scripts, training and evaluation entry points, baseline implementations, and machine-readable results. Fixed instances serialize topology, entity attributes, compatibility data, and distances needed for replay. Manifests record dataset, checkpoint, source, protocol, and result SHA-256 hashes where applicable, together with command-line settings and candidate seeds.

Principal seeds are 20260704 for the held-out generator, 20260720 for transfer derivation, 20260706–20260708 for training, 20260609 for exact solvers, and 0–31 for candidate sampling. The final 160,500-transition checkpoint is evaluated without validation- or test-based selection. The benchmark and implementation will be released publicly upon publication.

F.2 Computation

Training uses one RTX 4090 per run. Approximate GPU-hours per seed are 1.0 for HeteroMRTA, 1.1 for uniform selection, and 3.5 for TAGS. Evaluation allocates eight CPU cores to every method on the same AMD Ryzen 9 9950X. Appendix D.3 specifies candidate-generation timing, while Appendix C.2 specifies solver timing. The largest evaluated entity graph contains at most 105 valid entities.

F.3 Execution and action settings

Simulation uses static shortest paths, common unit speed, zero loading and service durations, and nondepleting storage. Deterministic service times can enter the completion updates through e_j. The work-conserving decoder chooses current service when available and otherwise moves to service-enabling storage. Delivery quantities are $\{1, 2, 3\}$; increasing $q_{\max}$ expands the conditional quantity vocabulary linearly.

The qualitative simulation in Figure 5 shows one 15-robot, 15-task rollout from D15. The quantitative study covers 15, 35, and 50 robots with equal task counts. The physical demonstration uses three wheeled robots with different capacities and collaborative capabilities configured in software, under online TAGS allocation. Together these views document the scheduling process at the evaluated simulation scales and its execution on the smaller physical platform.